Vorlesung Programmierung (Wintersemester 2013/2014)

1 Ziele der Vorlesung

- Vermittlung grundlegender Techniken zur Programmierung
- Gefühl für guten Stil entwickeln (Programme so schreiben, dass andere diese lesen können)
- Techniken zur Beherrschung komplexer Zusammenhänge (Abstraktion)
- Denken in Strukturen und Algorithmen
- Systematische Entwicklung von Programmen (Datenanalyse, Testfälle, Konstruktionsschemata)
- Grundkenntnisse verschiedener Programmierparadigmen (funktional, imperativ, objektorientiert)

2 Verwendete Programmiersprachen

Im wesentlichen **Scheme** (in der Syntax von **DrRacket**), wegen

- einfacher Syntax
- einfacher Umsetzung verschiedener Programmierparadigmen
- guter, frei verfügbarer Programmierumgebung (DrRacket)

3 Detaillierter Vorlesungsverlauf

25.10.: Einführung, Organisatorisches, grundlegende Begriffe wie Programmieren, Algorithmus, (partielle/totale) Korrektheit, Programmiersprache (Syntax, Semantik, Pragmatik), Stichworte zu Scheme, Arbeiten mit DrScheme

28.10.: Programmelemente: Ausdrücke (Zahlen, Kombination, Präfix-Notation), Abstraktion durch Benennung von Objekten (`define`), Umgebung, Termauswertung (informell), Berechnungsvorschriften und Prozeduren (`lambda`), Substitutionsmodell Kommentare

1.11.: Entwurfsrichtlinie „Kommentar vor Umsetzung“, Verträge, Signaturen, Sorten, Entwurfsrichtlinie „Signatur vor Umsetzung“, Testfälle (`check-expect`), Entwurfsrichtlinie „Testfälle vor Umsetzung“, Überdeckungswerkzeug von DrRacket, Entwurfsrichtlinie „Ausreichende Testfälle“, bedingte Ausdrücke (`cond`, `if`, `and`, `or`, `not`), Stringkonstanten, Sortenkonstruktion mit `one-of`, Beispiel Aggregatzustand von Wasser

4.11.: Konstruktionsschablone Fallunterscheidung, Quadratwurzelberechnung nach Newton, Testfälle `check-within`, top-down/bottom-up Entwurf, Bedeutung von Namen, lexikalische Bindung, Prozeduren und Prozesse, Fakultätsfunktion

8.11.: linearer rekursiver Prozess am Beispiel der Fakultätsfunktion, linearer iterativer Prozess am Beispiel der Fakultätsfunktion Baumrekursion am Beispiel der Fibonacci-Zahlen, Größenordnungen, O-Notation, Potenzrechnung (lineare und logarithmische Laufzeit)

11.11.: ggT (Euklidscher Algorithmus), Primzahltest durch Teilerprüfung, probabilistischer Primzahltest nach Fermat, Motivation zum Thema Datenabstraktion, Zusammengesetzte Daten am Beispiel von Schokokeksen

15.11.: Konstruktoren, Selektoren, Konstruktionsschablone „Zusammengesetzte Daten als Argumente“ Konstruktionsschablone „Zusammengesetzte Daten als Ausgabe“, Definition von Records (`define-record-procedures`), Beispiel rationale Zahlen, Operationen auf zusammengesetzten Daten Schichtenentwurf von Programmen, Module (`provide`, `require`)

18.11.: Definition gemischter Daten (`(signature (mixed ...))`), Konstruktionsschablone „Gemischte Daten als Argumente“, Darstellung von Listen am Beispiel von Zahlenlisten (Sorte `empty-list`, `empty`, `empty?`, Sorte `nonempty-list`, `cons`, `cons?`, `first`, `rest empty?`, `cons?`), Kasten-Zeiger-Darstellung, Listenoperation „Summe aller Elemente“, Konstruktionsschablone „Listen als Argumente“, String-Listen

22.11.: Definition beliebiger Listen, parametrisierte Records, Signaturkonstruktor, Signaturvariable, parametrisierte Paare, parametrisierte Listen in DrRacket, Listenoperation „Länge einer Liste“ und `concatenate`, Listenoperation `n-th`

25.11.: Baumstrukturen, Operation „Anzahl der Blätter“, symbolisches Differenzieren (Darstellung von Funktionsausdrücken und Implementierung ohne Vereinfachung von Funktionsausdrücken)

29.11.: Symbolisches Differenzieren mit Vereinfachung von Funktionsausdrücken, Schnittstelle für Mengen, Implementierung von Mengen durch ungeordnete Listen, Implementierung von Mengen durch geordnete Listen, Idee der Implementierung von Mengen durch geordnete Binärbäume

2.12.: Implementierung von Mengen durch geordnete Binärbäume Huffman-Bäume, Präfixcode, Idee der Konstruktion von Huffman-Bäumen, Datenstrukturen für Huffman-Bäume, Decodierung mit Huffman-Bäumen

9.12.: Konstruktion von Huffman-Bäumen, Prozeduren höherer Ordnung, Prozeduren als Parameter, lokale Definitionen (Sonderform `letrec`), Gültigkeitsbereich/Geltungsbereic
lexikalische Bindung, Benutzung der lexikalische Bindung, universelle Berechnungsverfahren auf Datenstrukturen `map-list`

13.12.: universelle Berechnungsverfahren (Nullstellenbestimmung durch Intervallhalbierung), Prozeduren als Ergebnis (Differenzierung von Funktionen, Funktionskomposition), Implementierung von Listen durch Funktionen höherer Ordnung
Berechnungen: Programmsignatur, Terme, Programm,

16.12.: Substitution, Position, Teilterm, Ersetzung, Berechnungsschritt, Normalform, strikte und nicht-strikte Auswertung
Abstrakte Datentypen: Schnittstelle, Gesetze

10.1.: Implementierung eines ADT, Beispiele rationale Zahlen und Listen. Einführung in die strukturelle Induktion Beweistechnik strukturelle Induktion (am Beispiel von `length`), Korrektheit von `append` mittels struktureller Induktion

13.1.: Eigenschaften, Sonderformen `for-all` und `==>`, Test `check-property`, Testen von ADT-Gesetzen am Beispiel rationaler Zahlen, Mengen und Listen
Veränderbare Zustände, Sonderformen `set!` und `begin`, Beispiel Abheben vom Bankkonto

17.1.: „Abhebungsprozessoren“ für mehrere Konten, Bankkonto mit Nachrichtenweitergabe für Abheben und Einzahlen, Diskussion der Probleme der Zuweisung (Substitutionsmodell nicht mehr anwendbar, Auswertungsreihenfolge relevant, referentielle Transparenz, Wertgleichheit vs. Objektidentität), Umgebungsmodell: Umgebung, Rahmen, Bindung

20.1.: Umgebungsmodell: Auswertungsregeln für Variablen und Kombinationen, Prozeduranwendung, lambda-Ausdrücke, `define`, `set!`, `letrec` und Konstruktorprozeduren; Beispiele für das Umgebungsmodell: Darstellung einfacher Prozeduren, lokale Zustände

24.1.: Beispiele für das Umgebungsmodell: lokale Definitionen im Umgebungsmodell.
Mutatoren, veränderbare Datenobjekte, Mutatoren für veränderbare Listen (`set-mfirst!`, `set-mrest!`), Implementierung veränderbarer Listen (`mcons`) durch Nachrichtenweitergabe und `set!` Umwandlungsprozedur `mlist->list`, Beispiel `append!`, Erzeugung zyklischer Strukturen, Mutatoren für beliebige Strukturen (`define-record-procedures-2`), Implementierung veränderbarer Bankkonten hiermit, Arbeiten mit mehreren veränderbaren Strukturen (Geldtransfer)

27.1.: Implementierung veränderbarer parametrischer Listen mit `define-record-procedures-paramet`
Structure Sharing und Identität, Bedeutung von `eq?`, Idee von Warteschlangen, Implementierung von Warteschlangen; Motivation zur Simulation digitaler Schaltkreise

31.1.: Simulation digitaler Schaltkreise, Schnittstellen, Schnittstelle von Drähten, Implementierung elementarer Gatter (Inverter, Und-Gatter, Oder-Gatter),

Implementierung von Drähten als Objekte, Zeitplan, Sonden, Simulationsablauf

3.2.: Zustände in nebenläufigen Systemen, Semaphore, Synchronisation von Zustandsänderungen mit Semaphoren, Verklemmungsproblem bei Semaphoren

7.2.: Bemerkungen zu unterschiedlichen Programmiermodellen; Syntaxbeschreibung mit EBNF
Zusammenfassung